NASA Technical Memorandum 83167

NASA-TM-83167 19810019289

REQUIREMENTS AND PRELIMINARY DESIGN FOR A GENERAL PURPOSE REAL-TIME EXECUTIVE FOR FLIGHT COMPUTERS

Kathryn A. Smith

JUNE 1981

NASA
National Aeronautics and Space Administration

Langley Research Center
Hampton, Virginia 23665

REQUIREMENTS AND PRELIMINARY DESIGN FOR A GENERAL PURPOSE REAL-TIME EXECUTIVE FOR FLIGHT COMPUTERS

SUMMARY

This document sets forth the requirements for a general purpose real-time executive for flight computers. It includes a description of and the requirements for the major elements of such an executive. These are an initialization module, a task scheduler, and an interrupt handler. It is intended that this document serve as a basic requirements document for a general purpose flight executive, independent of application and computer.

INTRODUCTION

Recent advances in computer technology have led to increasing application of digital control in all types of time critical processes. In NASA such digital systems are becoming more important in flight control. In flight computing, certain computations must be completed within given time intervals, data must be retrieved and processed on a regular basis, and interrupts must be handled without interfering with required computations. To accomplish these functions a real-time executive is needed to provide an interface between the applications programs and the flight hardware elements controlled by the computer.

This document outlines the requirements for a general purpose capability, allowing for differences in flight computers and application specifications through the use of system parameters and variables. It is evident that the development of a general purpose executive will reduce the cost of developing specific real-time flight executives, primarily by reducing duplication of effort. It is intended that this document serve as a basic requirements document for a general purpose executive, independent of application and computer.

DESCRIPTION OF EXECUTIVE

General Description

The major elements of this executive are (1) an initialization module to handle initialization and start-up of the system, (2) a task scheduler to ensure orderly and timely processing of both tasks and events, and (3) an interrupt handler to control the interaction of interrupts with the system. Figure 1 shows these major elements of the executive.

The executive also provides an interface between the computation process and the I/O devices. All processing by the executive will be controlled by priority levels; tasks and events will be scheduled by priority and interrupts handled by priority. This design limits the number of priority levels to 15.

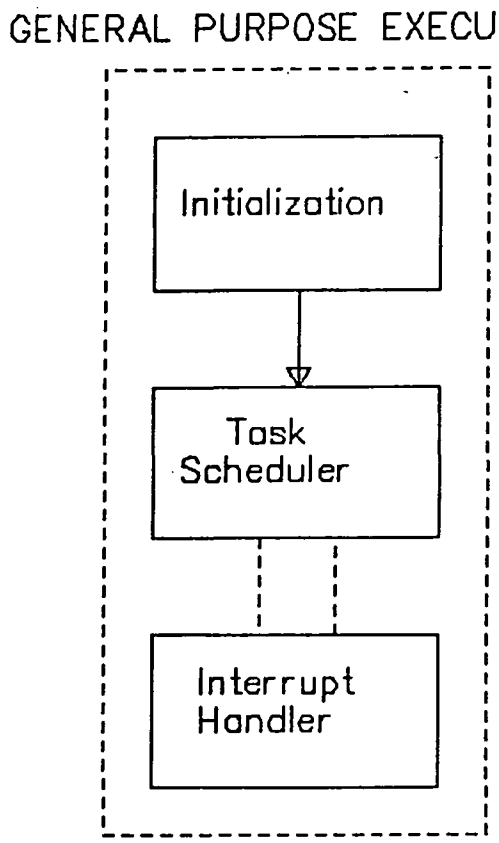

FIGURE 1 - MAJOR ELEMENTS of GENERAL PURPOSE EXECUTIVE

Tasks and events are computations performed by the flight computer according to the application specification. Task and event processes are program modules defined by the application user. Task processes occur periodically throughout the entire execution sequence. Event processes on the other hand are user defined modules that are executed when certain conditions occur, and these may be activated throughout the entire execution sequence or during some portion of it. Task and event modules may contain many computations and may in fact consist of a group of related tasks. Tasks and events are scheduled by priority from highest to lowest level.

The processing of interrupts is also controlled by priority levels. When an interrupt signal is received by the executive, control will be transfered to the interrupt handler. The interrupt handler will then determine if these interrupts should be processed immediately or should be put into a pending stack for later processing.

User defined program modules which perform tasks, events and interrupt processing are called processes. Each module will have an associated, user-defined priority level and a table in which to store the machine state if it

is interrupted. Processes may only be interrupted by interrupts that have a higher priority level. The number of task modules, the number of interrupts, and the number of events will be defined as system parameters. Impending processes are tasks, events, or interrupt modules which have been interrupted before they could complete processing.

The executive will handle up to 15 priority levels. Level 1 is the highest priority level and level 15 is the lowest. The first four levels will be reserved for the executive and will not be available to the user. These levels will be reserved as follows: level 1 - power off interrupt, level 2 - machine fault interrupt, level 3 - interval timer interrupt, and level 4 - minor cycle and fast tasks. The user may assign the remaining eleven levels as desired. Table 1 summarizes the assignment of priority levels.

Table 1

Interrupts and Tasks by Priority Level

Priority	Interrupt	Task	Event
1	Power off		
2	Machine fault		
3	Interval timer		
4		Minor cycle	
5-15	user set	user set	user set

Most computations are performed periodically within some cycle or time frame. Cycle lengths are application dependent and can include a minor (or fast) cycle, intermediate cycles, and a major cycle. Each computation cycle will be assigned a priority level. The minor cycle will have the highest priority level (level 4), followed by the intermediate cycles and the major cycle. All tasks with a given priority will be grouped into a task module which will be given the same priority and will be associated with the computation cycle with the same priority.

The minor (or fast) cycle is the driving computation cycle of the executive. It is machine and application dependent and is controlled by the hardware interval timer. Generally the interval timer will be driven by an I/O interrupt indicating that input is ready. The length (time) of the minor cycle will be stored as a system parameter. Each minor cycle will be initiated by an interval timer interrupt. A task complete flag will be provided for the minor cycle tasks. This flag will be tested at the beginning of every minor cycle to ensure that all fast tasks were completed within the last minor cycle. If these tasks did not complete processing, an error condition exists and control will be given to the error module.

The intermediate and major cycles will be defined as multiples of the minor cycle, although these cycles need not be multiples of each other. The major cycle will be defined as the longest, or slowest, cycle. The length of each intermediate cycle and the major cycle will be set by an associated cycle counter. The user will be allowed to define up to 10 intermediate cycles and

one major cycle. The number of intermediate cycles will be stored as a system parameter.

Counters will be maintained for all intermediate cycles and the major cycle. Cycle counters will be set to their maximum values during initialization and decremented every minor cycle. The counter for a cycle, the maximum count, and a cycle complete flag will be stored in a table with entries for each associated task module. The task counters will be tested every minor cycle. When a task completes processing, its task complete flag is set. Upon completion of an intermediate cycle (cycle counter = 0), the tasks for this cycle should be complete, and the task complete flag will be tested. If the task complete flag is on, the counter will be reset to its maximum, the task complete flat reset, and that cycle restarted. If the task complete flag has not been set, an error condition exists (tasks for this cycle are incomplete) and control is given to the error module. Figure 2 shows this cycle count test for cycle i.

Counters will also be maintained for periodic events. These counters will be set to initial values during initialization and decremented every minor cycle. These event counters will be tested every minor cycle to determine which event(s) should be processed. An event will be processed when the associated counter is zero and the event is enabled. Upon completion of an event its associated event counter will be reset.

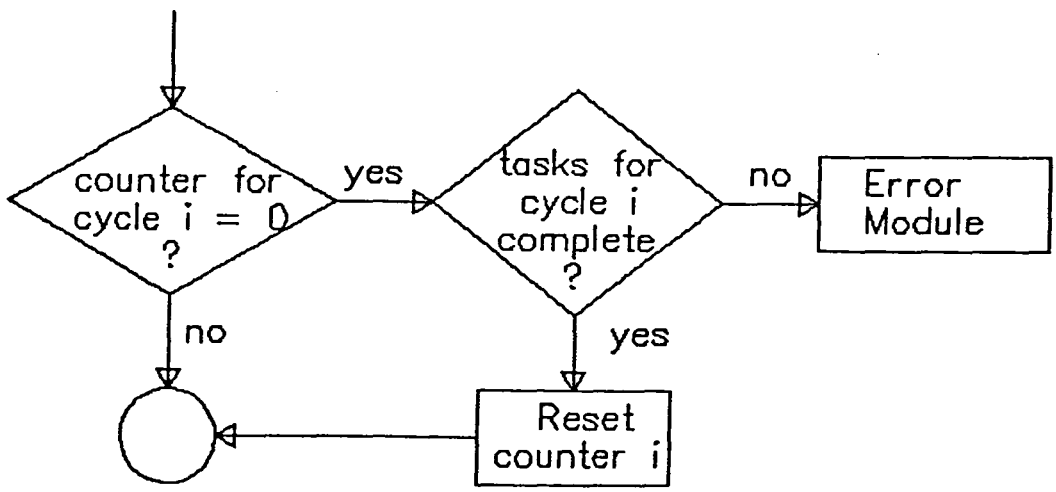

FIGURE 2 - Cycle Count Test

Initialization

After the flight computer is powered on, the loading process will begin. A bootstrap loader is one possible way to handle this (see Appendix B). The loading process will include the initialization of memory and of the registers, most likely either to zero or to some preset system value. Following initialization, control will be given to the executive which will initialize the task and event counters, start the interval timer and the real-time clock, and pass control to the task scheduler.

Task Scheduler

An important function of the executive is the task scheduler, which schedules tasks and events to make efficient use of minor cycle time. Some tasks and events must be processed more frequently than others. Also, certain time critical tasks and events must be completed periodically within the shortest cycle known as the minor or fast cycle. Others need only be completed within the major cycle. So, all task and event processes must be managed to make best possible use of the computation time available during each minor cycle.

The task scheduler will handle the processing of tasks, timed events, and pending interrupts and ensure that tasks are completed within their allotted cycles. Tasks will be scheduled as follows: minor cycle (fast) tasks, foreground tasks from highest to lowest priority, and background tasks. Background tasks have the lowest priority and are scheduled only on a time available basis within the major cycle. Timed events will be scheduled by priority level, before tasks of the same level. The user may optionally indicate to the executive whether tasks and/or events are to be scheduled. The task scheduler will also keep track of the various cycles by means of cycle counters. Pending interrupts will be processed after the minor cycle tasks and before remaining tasks and events of equal or lower priority level. This can be seen in Figure 3.

Tasks

Tasks are computations performed by the flight computer, and task processes are application program modules defined by the user to perform these tasks. They occur periodically throughout the entire execution sequence. Associated with each task process will be an entry in the task table containing such information as the transfer address and cycle count. When a task is scheduled, control will be passed from the executive to the task module by means of a transfer address in the task table. Processing of the task module will continue until the task completes or until the task is interrupted by a process which has higher priority. This can be seen in Figure 4.

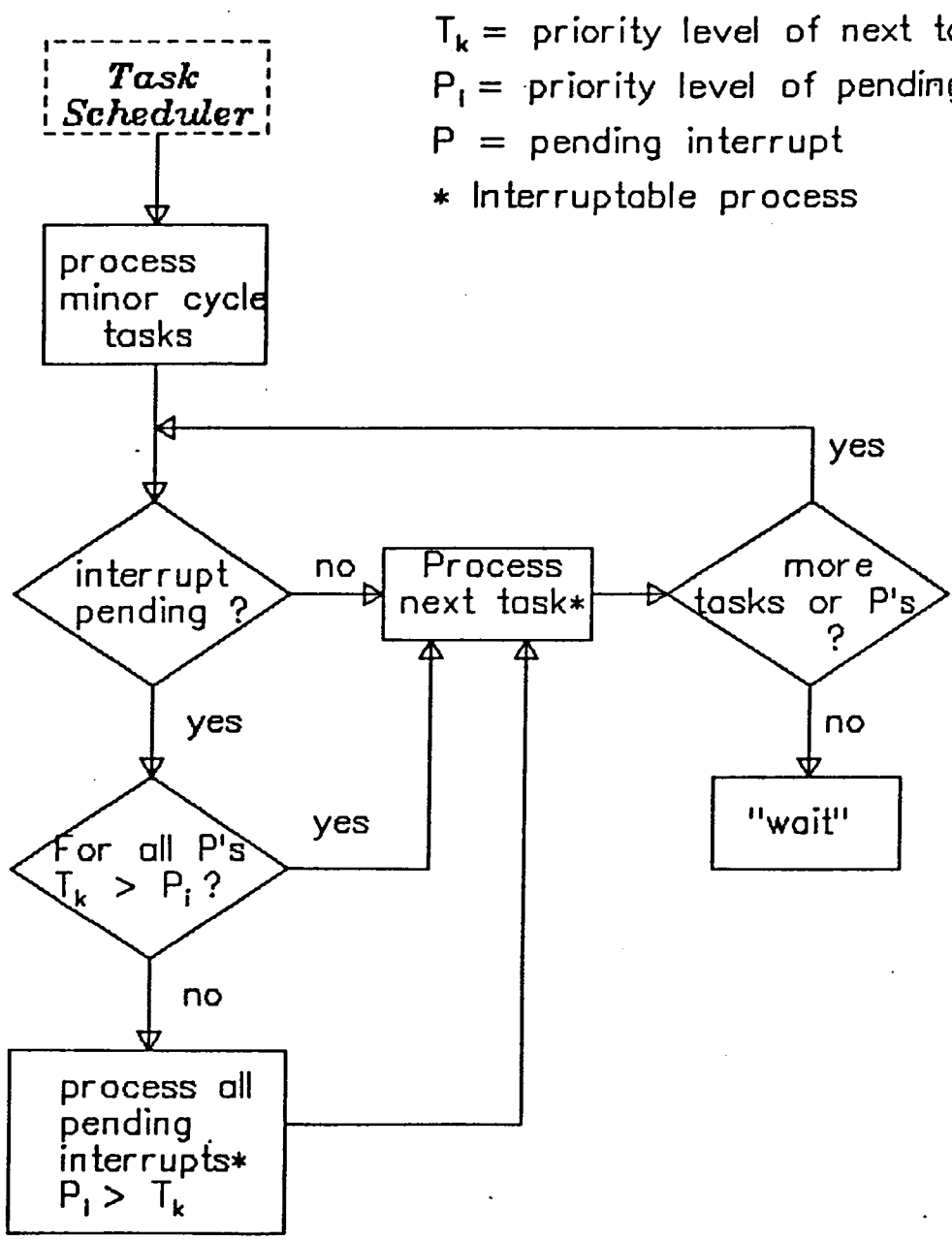

FIGURE 3 - Task Scheduler

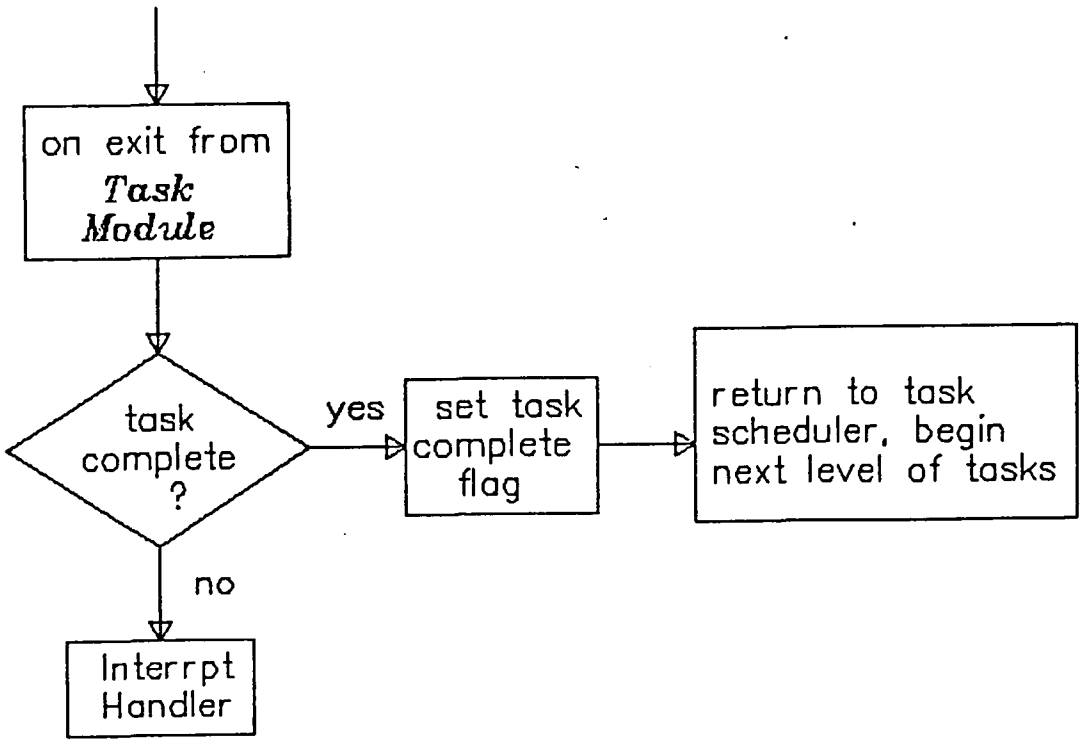

FIGURE 4 - EXITTING FROM T4SK PROCESS

Several priority levels of tasks will be allowed. Each priority level will have an associated task process, which can contain any number of computations. The number of tasks and the priority of each will be set by the user. Tasks may be interrupted by higher priority level interrupts.

Minor cycle processing: Minor cycle processing will include all tasks to be processed each minor cycle. These are time critical tasks which have the highest priority level allowed to the user and, thus, are not interruptable by the user.

Foreground Tasks: Foreground tasks are those intermediate tasks which have priorities lower than the minor cycle tasks and higher than background tasks. Each intermediate cycle will have an associated task module (set of tasks) to be completed within that cycle.

Background Tasks: Background tasks are low level tasks which are processed on a time available basis. Background tasks must be completed within the major cycle.

Task Scheduling: Tasks are scheduled in the following order: minor cycle, foreground (from highest to lowest priority), and background. Following completion of the fast tasks, foreground tasks will be scheduled by priority level. Tasks of priority n+1 are scheduled upon completion of level n tasks. The scheduler will transfer control to the next lowest level which has not completed processing--that is, the task scheduler can either restart an interrupted task or initiate a new one. It will be the user's responsibility

to ensure that tasks are completed within the proper time frame. Upon completion of a task process, the task complete flag will be set and processing of the next level of tasks initiated. If all tasks complete processing before the end of the major cycle, the executive will enter a "wait" state. An interval timer interrupt will take the executive out of this wait state. Figure 5 illustrates the scheduling of tasks over several minor cycles. In this example there are four levels of tasks--minor cycle (fast task), task 1, task 2, and background--and one event, which has the same priority level as task 2. Task 1 repeats every 2 minor cycles and thus must be complete within two minor cycles, and task 2 repeats every five minor cycles. Since the event has the same priority level as task 2, it will be scheduled first as seen in cycle 6. In this example, the length of the major cycle is 10 minor cycles. Also note that the executive is in a wait state at the end of cycle 10.

Interrupting Tasks: A task can be interrupted by processes which are of a higher priority level. When a task is interrupted the following must be saved in the machine state table: contents of the registers, and the address of the next instruction to be executed. A pointer to the machine state table will be saved for this task in the interrupted process stack.

Restarting Tasks: When control is returned to the executive following an interrupt, the task processes will be restarted as follows: If an interval timer interrupt has occurred, the fast tasks will be initiated. Upon completion of the fast tasks, the executive will activate the process with the highest priority level. If the cycle counter for that level is at its maximum, the executive will initiate that level of tasks. Otherwise, the executive will reload the registers from the machine state table and activate the previously interrupted task from the transfer address in the table. In the case of lower level interrupts, other than those transferring to the error module, the machine state for the last interrupted task will be reloaded and that task resumed.

Events

Events are computations performed when special conditions occur, and event processes are user defined program modules to perform them. Events may be scheduled to occur during the entire execution sequence or during a portion of it. Events may be specified in several ways. They may be (1) timed, (2) external, or (3) background. Associated with each timed event process will be an event table containing information pertinent to its execution. The form of the table will depend on the type of event.

Timed events may be specified in two ways: (1) as a function of the real-time clock, and (2) periodic. Timed events differ from tasks in that they can be enabled and disabled. Thus it is possible for an event to function only during some small time period of the complete execution cycle.

A clock event occurs at a specified time based on the real-time clock. During every minor cycle after completion of the fast task, the times for clock events will be tested against the real-time clock. A clock event will

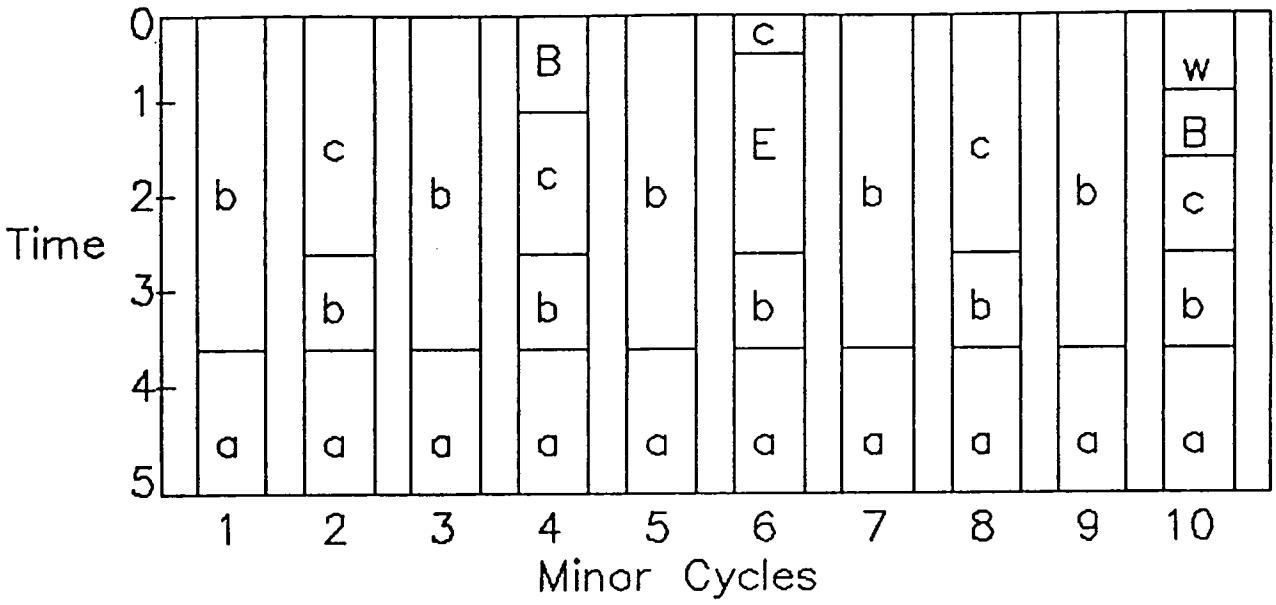

FIGURE 5 – Example of Task Scheduling

be scheduled when the real-time clock has reached the event time. If a clock event is scheduled, control will be passed to the clock event module. Control will return to the executive upon completion of the event.

A periodic event occurs periodically based on the interval timer. During every minor cycle the executive will test the counters in the cyclic events table. Any event with a counter of zero, and which is enabled, will be scheduled. Note that more than one event can be scheduled in a minor cycle, and this will be on a first-come-first-served basis. After an event is processed, its associated counter will be reset to its maximum.

External events are activated by an external interrupt. These generally have a one time occurrence. External events will be initiated from the user defined external event interrupt module.

Background events are low level periodic events that are processed after all other tasks and events are processed. They are processed during any spare time in the major cycle; if no time is available, background events will be ignored.

Interrupting Events: An event can be interrupted by interrupt processes which are of a higher priority level. When an event is interrupted the following must be saved: machine state (registers) in the machine state table, the address of the next instruction to be executed, and a pointer (in the interrupted process stack) to the entry in the machine state table (see Appendix A).

Restarting Events: When an event is restarted, the executive will reload the registers and machine state from the associated machine state table and transfer to the appropriate address.

Interrupt Handler

When an interrupt signal is received by the executive, control will be transferred to the interrupt handler. The interrupt handler will determine if the interrupts should be processed immediately or should be put into a pending stack for later processing. Interrupts will be defined in terms of priority level and type. This information will be stored in the interrupt table (see Appendix A). If interrupts occur concurrently, then each of the interrupts will be recognized and processed by priority level, i.e., highest level first. The lower level interrupts will be put into a pending stack for later processing. Each type of interrupt will have an associated interrupt process module. Interrupt processes, including those for high level interrupts, are user defined modules to handle each of the interrupts. Low level interrupt processes may be interrupted by higher level interrupts. Figure 6 shows the basic form of the interrupt handler.

If the priority level of the new interrupt is greater than the level of the currently active process, the present machine state will be saved and a pointer to the machine state table added to the interrupted process stack. Control will then be passed to the new interrupt process. Before restoring

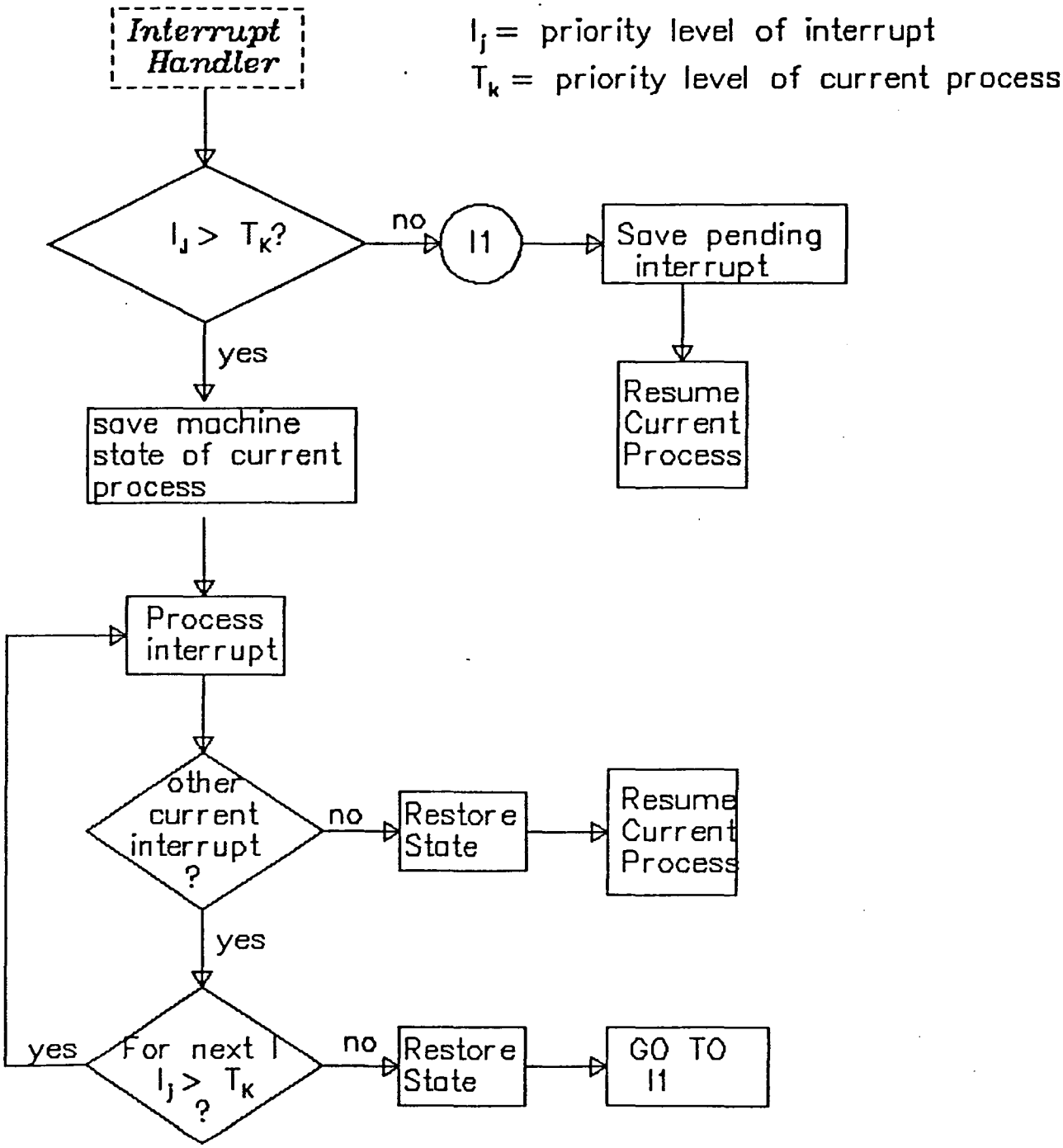

FIGURE 6 - Interrupt Handler

the interrupted state, the interrupt handler will activate any other current interrupt whose level is greater than the level of the interrupted process. The pointers in the interrupted process stack will be used to restart the interrupted processes.

If the level of the new interrupt is not greater than that of the current process, the interrupt will be added to the pending interrupt stack for later processing and the pending interrupt flag (a system variable) incremented. Pending interrupts are low level interrupts which have been recognized, but because their priorities are lower than the level of the current task have been allowed to begin processing. Interrupts on the pending stack will be initiated at a later time by the task scheduler, according to priority.

After an interrupt is recognized, if the priority of the current process is higher than the priority of the interrupt, the following information will be saved in a pending interrupt stack:

priority level
type of interrupt
transfer address
buffer protect flag

After each task completes processing, the pending interrupt stack is interrogated. If any interrupts are pending, then those pending interrupts which are of equal or higher priority level than the next task to be scheduled will be processed. The pending interrupt flag will be decremented by the number of pending interrupts honored.

Interrupt Priority Levels: All interrupts and their associated interrupt processes will be assigned priority levels. The highest level interrupts will be reserved for the executive. User set priorities will be below those reserved for the system. The user will be responsible for selecting and setting the priority levels of the various interrupt processes.

Types of Interrupts: The general purpose executive allows for several types of interrupts including the following: power off, machine fault, external, and arithmetic fault. The system will also allow for "other" user defined interrupts. The priorities of the power off, machine fault and interval timer interrupts are pre-defined. All other interrupts will have user set priorities. The interrupt modules may be interrupted by higher level interrupts.

The Power Off Interrupt is the highest level of interrupt. If a power failure occurs, control will be transferred immediately to the power off module. The power off module will accomplish all that is necessary for an orderly shutdown of the system.

The Machine Fault Interrupt is a high level interrupt which signals a hardware failure. When the executive receives this interrupt, control will be transferred to a user defined error module for special processing.

External Interrupts are signals from external devices, such as (1) the interval timer, (2) I/O device, (3) external events, (4) external clock, and (5) other user defined external interrupts. These are discussed in further detail below.

(1) Every minor cycle an interval timer interrupt will give control to the executive. It signals the beginning of a new minor cycle. Control will be passed from the interval timer module back to the executive. At this point the executive will schedule the fast (minor cycle) tasks, and then decrement the task and event counters. An interval timer interrupt will supersede all other interrupts except the power off and machine fault.

(2) An I/O device interrupt signals that I/O is ready for the system. A typical I/O interrupt signals that an input buffer has been filled and is waiting for processing. Processing of this interrupt will be done by the I/O Interrupt Module. Upon completion of this, control will return to the interrupt handler where the interrupted task will be restarted.

(3) An External Event Interrupt indicates that an external event should be processed. The external event will be initiated if it is of higher priority than the current process; otherwise it will be added to the pending interrupt stack. After processing the event, control will be returned to the interrupt handler and the interrupted task restarted.

(4) The External Clock Interrupt allows a clock other than the interval timer and real-time clock to interrupt processing.

(5) An option will be included to allow for user defined external interrupts which would not be included in one of the other external interrupts.

The Arithmetic Fault Interrupt is a relatively low level interrupt. It indicates that some anomalous arithmetic operation, such as division by zero, has occurred. This interrupt causes control to be passed to a user defined error module.

An option will be included to allow the user to define other interrupts which have not been specifically defined.

CONCLUDING REMARKS

This paper defines the requirements for a general purpose, real-time executive for flight computers, which allows for differences in flight computers through the use of system parameters. The major elements of this general purpose executive are (1) an initialization module to handle initialization and start up of the system, (2) a task scheduler for both tasks and events, and (3) an interrupt handler to control the interaction of interrupts with the system. The executive outlined in this paper has been reviewed with respect to its use in a number of applications. This document is presently being used as the requirement specifications for the design of the flight executive for the LaRC supported Terminal Controlled Vehicle (TCV) Project.

Langley Research Center
National Aeronautics and Space Administration
Hampton VA 23665
June 19, 1981

APPENDIX A

TABLES AND STACKS

The general purpose executive, in the performance of its duties (task and event scheduling, and interrupt processing), will use several tables and stacks containing information necessary to perform these functions. Tables will be used for interrupts, events, tasks, and interrupted processes. Stacks will be used for pending interrupts and interrupted process pointers. These tables and stacks are discussed in detail below.

Interrupt table: This table will be used to control the interrupt handler. Information about interrupts will be stored from highest to lowest priority level. This table will have an entry for every interrupt and each table entry will contain at least the following information.

Priority level
Type
On/off flag
Interrupted flag
Transfer address
Buffer protect flag

Timed Event tables: There are two types of timed events which will require tables. They are periodic events and clock events. Event tables will contain at least the following information:

(a) Periodic event table (interval timer)
Counter (to be decremented)
Max count
Transfer address
Enable/disable flag
Interrupted flag

(b) Clock events (real time clock)
Event time
Transfer address
Enable/disable flag

Task table: The table will be used by the task scheduler to schedule tasks. This table will have an entry for each task module and will include at least the following information for each entry:

Cycle counter (to be decremented)
Maximum cycle count
Task complete flag
Pointer to current address (restart)
Pointer to beginning address
Priority level

APPENDIX A

Machine state table: Each module or process (task, event, interrupt) will have an entry in this table, containing the last contents of the registers (last state of the system) for each module and the return address. This table will be used during restart procedures.

Pending interrupt stack: This stack will be used to indicate pending interrupts. It should include at least the following information for each pending interrupt:

Priority of interrupt
Type of interrupt
Transfer address

Interrupted process stack: This stack will contain pointers to interrupted processes in the machine state table and will be used during restart procedures.

APPENDIX B

GLOSSARY

Bootstrap loader: A load process in which the first few instructions of a program are loaded into memory and then those instructions are used to load the rest of the program.

Cycle: Period or time frame during which computations are performed.
- Minor (fast) cycle: The shortest computation cycle which is the driving cycle of the executive.
- Intermediate cycle: Computation cycle defined as a multiple of the minor cycle.
- Major cycle: The longest, or slowest, computation cycle.

Cycle counter: A counter which is used to keep track of all intermediate cycles and the major cycle in terms of the minor cycle.

Events: Computations which are performed under special conditions by the flight computer.
- Timed events: Events activated by some time element.
- Clock event: Events which occur at a specified time based on the real-time clock.
- Periodic event: Events which occur periodically based on the interval timer.
- External events: Events activated by an external interrupt.
- Background events: Low level periodic events that are processed after all other tasks and events are processed.

External clock: A clock external to the executive.

Interval timer: A timer which gives a periodic signal every minor cycle.

Machine state: The contents of the registers at any given moment.

Priority: A pre-set number used to determine the order for processing events, tasks and interrupts.

Process: A process is a user defined program module which performs tasks, events, or interrupt processing.

- Task process: User defined program module which performs tasks periodically throughout the execution sequence.
- Event process: User defined program module that is executed when certain conditions occur, and this may be activated throughout the entire execution sequence or during some portion of it.
- Interrupt process: User defined program module to handle one of the interrupts.

APPENDIX B

Real-time clock: A timer which gives the real-time from initialization of the executive.

Schedule: To initiate or restart a task or event process.

Tasks: Computations which are performed by the flight computer periodically throughout the execution sequence.

Minor cycle (fast) tasks: Time critical tasks which must be processed each minor cycle.

Foreground Tasks: Tasks of lower priority than the fast tasks, performed during the intermediate cycles.

Background (major cycle) Tasks: Background tasks are low level tasks which are processed on a time available basis. They must be completed within a major cycle.

REFERENCES

1. Viking Flight Program Description Document, Martin Marietta Corporation, Denver, Colorado, 83767000130, Vol I., 1973.

2. AGS Software Requirements Document, Sperry Flight Systems Division, Sperry Rand Corporation, File No. 5121.8085.14.6-1, June 1979.

3. Cleveland, J. I., Crawford, D. J., and Rowell, L. F., Reference Manual for the Langley Research Center Flight Simulation Computing System, NASA Langley Research Center, NASA TM 78757, June 1978.

4. Margolis, S. P., Wolverton, D. A., and Hamm, R. L., Flight Control Computer Software Description, Computer Sciences Corporation, Hampton, Virginia, CSC Number 33508, July 1979.

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
NASA TM-83167		

4. Title and Subtitle	5. Report Date
Requirements and Preliminary Design for a General Purpose Real-Time Executive for Flight Computers	June 1981
	6. Performing Organization Code 506-61-43-05
7. Author(s) Kathryn A. Smith	8. Performing Organization Report No.
	10. Work Unit No.
9. Performing Organization Name and Address NASA Langley Research Center Hampton VA 23665	11. Contract or Grant No.
	13. Type of Report and Period Covered Technical Memorandum
12. Sponsoring Agency Name and Address National Aeronautics and Space Administration Washington DC 20546	14. Sponsoring Agency Code

15. Supplementary Notes

16. Abstract

This document provides the requirements and preliminary design of a general purpose, real-time executive computer program for flight computers. The paper includes a description and the requirements for the major elements of the executive: an initialization module, a task scheduler, and an interrupt handler. Task priorities and various process interrupts are discussed. System parameters and variables are identified which make the design adaptable to various flight computer and application specifications.

17. Key Words (Suggested by Author(s))	18. Distribution Statement
Real-time executive program Flight computers Flight software General purpose executive program	Unclassified - unlimited Subject Category 61

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price*
Unclassified	Unclassified	20	A02

For sale by the National Technical Information Service, Springfield, Virginia 22161

End of Document